

PR #54299 完整报告

vllm-project/vllm

[Perf] Avoid h2d copies from non-pinned CPU tensors

合并时间: 2026-08-29 12:06

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54299>

执行摘要

- 一句话: 修复非 pinned CPU 张量 H2D 拷贝导致的同步 stall
- 推荐动作: 值得精读。重点看两个设计模式: 一是 FlashInfer 中“拷贝前 pin_memory() 快照”如何在 MRV2 无显式同步的约束下同时满足异步与防竞态, 替代常驻 pinned 双缓冲; 二是 rot_pos_emb / rotary_pos_emb_thw 通过调整 get_cos_sin 调用顺序、直接在 GPU 设备上构造索引张量来彻底消除 H2D。这两个模式可在后续新增推理路径时直接复用。

功能与动机

PR body 明确说明动机: "This can cause a stall that impacts overlap of gpu and cpu work. These instances were exposed by <https://github.com/vllm-project/vllm/pull/53491>." 在 PyTorch 中, 从非 pinned CPU 张量发起的 `copy(..., non_blocking=True)` 无法真正异步, 会隐式同步并阻塞 GPU 侧推进; PR #53491 为相关路径启用异步拷贝后, 这些“假异步”点被暴露出来。本 PR 的目标是让 H2D 拷贝要么真正异步 (先 pin 再拷贝), 要么彻底不发生 (直接在目标设备上构造张量)。

实现拆解

1. FlashInfer 注意力元数据路径: pin 快照替代常驻 pinned 双缓冲 (`vllm/v1/attention/backends/flashinfer.py`) - 删除 `_make_buffer()` 方法及 `self.pin_memory` 属性 (原逻辑为 `not vllm_config.use_v2_model_runner and PIN_MEMORY`)。 - `paged_kv_indptr`、`paged_kv_last_page_len` 改为显式 `pin_memory=False` 的 `CpuGpuBuffer`; `paged_kv_indices` 改为常驻 GPU 的 `torch.zeros`, `_get_decode_wrapper()` 中同步去掉 `.gpu` 访问。 - `_compute_flashinfer_kv_metadata()` 删除 `paged_kv_indptr_cpu_buffer` 双缓冲, 改为拷贝前对 `paged_kv_indptr_cpu[:num_reqs + 1]` 执行 `pin_memory()` 快照, 再 `copy(non_blocking=True)`。 - 原因: MRV2 无显式同步, 常驻 pinned 缓冲存在 step N 异步拷贝与 step N+1 写入的竞态; 拷贝前快照一次解决“可异步”与“防竞态”两个问题。
2. 权重加载上下文恢复为 pinned 张量 (`vllm/model_executor/model_loader/utils.py`) - `device_loading_context()` 将 `original_device_states: dict[str, torch.device]` 改为 `cpu_params: set[str]`。 - finally 恢复 CPU 参数时由 `p.data.to(original_device)` 改为 `torch.empty_like(p.data, device="cpu", pin_memory=use_pin_memory).copy_(p.data)`; UVA 再 offload 路径也由两次拷贝 (`.to("cpu").pin_memory()`) 合并为一次 pinned 拷贝。 - 影响: 权重 offloading / sleep-wake 后回到 CPU 的参数即为 pinned, 后续加载或

H2D 不再引入同步 stall。

3. Logprobs 输出张量直接 pinned (vllm/v1/outputs.py) - LogprobsTensors.empty_cpu() 中 logprob_token_ids、logprobs、selected_token_ranks 均以 pin_memory=PIN_MEMORY 创建，并改用 new_empty 从 logprob_token_ids 派生，保证同源同设备。
4. 旋转位置编码：在目标设备上直接构造索引 (qwen3_omni_moe_thinker.py、qwen2_5_vl.py、ernie45_vl.py、glm4_1v.py、qwen2_vl.py) - 以 rot_pos_emb() / rotary_pos_emb_thw() 为例：把 get_cos_sin() 提前到构造 pos_ids 之前，torch.arange(h/w, device=cos.device) 直接在 GPU 上生成索引，删除原先 pos_ids.to(cos.device, non_blocking=True) 的 CPU→GPU 拷贝。
5. 多模态 / 音频编码器掩码走 async_tensor_h2d (phi4mm_audio.py、idefics2_vision_model.py、qwen3_asr.py、layers/pooler/tokwise/methods.py) - enc_streaming_mask 等由 contiguous().to(device, non_blocking=True) 改为 async_tensor_h2d(...)，由工具函数统一处理“pinned 则异步、非 pinned 先 pin 或安全回退”。
6. 测试与配置配套 - 本 PR 未新增测试文件；行为受 PIN_MEMORY (是否启用拷贝前 pin 快照) 与 VLLM_WEIGHT_OFFLOADING_DISABLE_PIN_MEMORY (模型加载上下文恢复是否 pinned) 两个开关控制。

关键文件：

- vllm/v1/attention/backends/flashinfer.py (模块 注意力后端；类别 source；类型 core-logic；符号 _make_buffer, _compute_flashinfer_kv_metadata, _get_decode_wrapper) : V1 引擎 FlashInfer 注意力元数据构建主路径：移除 _make_buffer/self.pin_memory 常驻 pinned 双缓冲，改为拷贝前对源切片 pin_memory() 快照；paged_kv_indices 改为常驻 GPU 的 torch.zeros。这是本次改动中风险最高、收益最直接的推理热路径。
- vllm/model_executor/model_loader/utils.py (模块 模型加载；类别 source；类型 data-contract；符号 device_loading_context) : 设备加载上下文是权重加载 /offloading 公共路径：恢复 CPU 参数与 UVA 再 offload 时改为直接分配 pinned CPU 张量一次拷贝完成，避免非 pinned 张量导致后续 H2D stall。
- vllm/model_executor/models/phi4mm_audio.py (模块 模型实现；类别 source；类型 data-contract；符号 forward_embeddings, calculate_hs_mask) : Phi-4 音频 Conformer 编码器两处 streaming mask 的 H2D 改为 async_tensor_h2d (内部处理 pin + non_blocking)，是音频编码路径上的直接收益点。
- vllm/v1/outputs.py (模块 输出处理；类别 source；类型 dependency-wiring；符号 empty_cpu) : LogprobsTensors.empty_cpu 创建 pinned CPU 张量，输出 logprobs 每 step H2D 拷贝因此可真正异步，影响所有请求的输出后处理路径。
- vllm/model_executor/models/qwen3_omni_moe_thinker.py (模块 模型实现；类别 source；类型 data-contract；符号 rot_pos_emb) : Qwen3-Omni 视觉编码器 rot_pos_emb 重排 get_cos_sin 调用顺序，直接在 cos.device 上构造 arange，消除 pos_ids 的 CPU 构造 + H2D，是“消除而非异步化”的代表。

- `vllm/model_executor/models/qwen2_5_vl.py` (模块 模型实现; 类别 source; 类型 data-contract; 符号 `rotary_pos_emb_thw`) : Qwen2.5-VL 与 Qwen3-Omni 相同模式: `rotary_pos_emb_thw` 的 `arange` 直接建在 `cos.device` 上, 去掉中间 H2D, 覆盖最大用户群体的视觉模型。
- `vllm/model_executor/models/idefics2_vision_model.py` (模块 模型实现; 类别 source; 类型 data-contract) : Idefics2 视觉模型引入 `async_tensor_h2d` 替换非 pinned H2D 路径 (结构性信号为导入关系调整), 属于多模态模型统一改造的一部分。
- `vllm/model_executor/models/qwen3_asr.py` (模块 模型实现; 类别 source; 类型 data-contract) : Qwen3-ASR 音频编码路径引入 `async_tensor_h2d` 统一处理 pin + 非阻塞拷贝, 与 `phi4mm_audio` 同属音频编码器改造。
- `vllm/model_executor/models/ernie45_vl.py` (模块 模型实现; 类别 source; 类型 data-contract) : ERNIE4.5-VL 旋转位置编码 / 掩码构造路径消除非 pinned H2D (结构性信号为配置键调整), 属多模态统一改造。
- `vllm/model_executor/models/glm4_1v.py` (模块 模型实现; 类别 source; 类型 data-contract) : GLM-4.1V 与 PR label 'glm' 对应, 位置编码路径同模式调整, 属于标签覆盖的模型族。
- `vllm/model_executor/models/qwen2_vl.py` (模块 模型实现; 类别 source; 类型 data-contract) : Qwen2-VL 与 Qwen2.5-VL 同族, 位置编码路径同模式调整, 属 qwen 标签覆盖的模型族。
- `vllm/model_executor/layers/pooler/tokwise/methods.py` (模块 池化层; 类别 source; 类型 data-contract) : tokwise 池化方法引入 `async_tensor_h2d` (结构性信号为导入关系调整), 保证池化路径的 H2D 也走统一 pin 逻辑。

关键符号: `_compute_flashinfer_kv_metadata`, `_make_buffer`, `device_loading_context`, `empty_cpu`, `rot_pos_emb`, `rotary_pos_emb_thw`, `forward_embeddings`, `calculate_hs_mask`

关键源码片段

`vllm/v1/attention/backends/flashinfer.py`

V1 引擎 FlashInfer 注意力元数据构建主路径: 移除 `_make_buffer/self.pin_memory` 常驻 pinned 双缓冲, 改为拷贝前对源切片 `pin_memory()` 快照; `paged_kv_indices` 改为常驻 GPU 的 `torch.zeros`。这是本次改动中风险最高、收益最直接的推理热路径。

```
# Preparing persistent buffers
# 关键设计: MRV2 下不复用常驻 pinned CPU buffer,
# 避免 step N 的异步 H2D 与 step N+1 的 buffer 更新产生竞态;
# 改为在每次拷贝前对切片做 pin_memory() 快照。
self.paged_kv_indptr = CpuGpuBuffer(
    max_num_reqs + 1,
    dtype=torch.int32,
    device=self.device,
    pin_memory=False,
)
# 原先的 CpuGpuBuffer 双镜像改为纯 GPU 常驻,
# 由 _copy_page_indices_ker 在 GPU 侧直接改写。
```

```

self.paged_kv_indices = torch.zeros(
    max_num_pages,
    dtype=torch.int32,
    device=self.device,
)
self.paged_kv_last_page_len = CpuGpuBuffer(
    max_num_reqs,
    dtype=torch.int32,
    device=self.device,
    pin_memory=False,
)

# ---- 以下位于 _compute_flashinfer_kv_metadata() ----
# 每次 step 将 indptr 拷到 GPU: 非 pinned 张量做 non_blocking
# H2D 会退化为同步并 stall, 因此先 pin_memory() 生成快照。
# 快照同时隔离了 cuda graph 模式下 CPU 侧被后续步骤改写的问题,
# 语义等价于被删除的 paged_kv_indptr_cpu_buffer 双缓冲。
paged_kv_indptr = self.paged_kv_indptr.gpu[: num_reqs + 1]
paged_kv_indptr_cpu = self.paged_kv_indptr.cpu[: num_reqs + 1]
if PIN_MEMORY:
    paged_kv_indptr_cpu = paged_kv_indptr_cpu.pin_memory()
paged_kv_indptr.copy_(paged_kv_indptr_cpu, non_blocking=True)

# write self.paged_kv_indices inplace
num_actual_pages = self.paged_kv_indptr.np[num_reqs]
paged_kv_indices = self.paged_kv_indices[:num_actual_pages]

```

vllm/model_executor/model_loader/utils.py

设备加载上下文是权重加载 /offloading 公共路径: 恢复 CPU 参数与 UVA 再 offload 时改为直接分配 pinned CPU 张量一次拷贝完成, 避免非 pinned 张量导致后续 H2D stall。

```

@contextmanager
def device_loading_context(module: torch.nn.Module, target_device: torch.device):
    """把 CPU 上的参数临时搬到 target_device 执行后处理,
    结束后恢复回 CPU; 恢复时直接写入 pinned 内存。"""
    if target_device.type == "cpu":
        yield module
        return

    cpu_params: set[str] = set()
    uva_offloaded_parameters: list[str] = []

    # 记录哪些参数原本在 CPU, 并临时搬到 GPU
    for name, p in module.named_parameters():
        if p.device.type == "cpu":
            cpu_params.add(name)
            p.data = p.data.to(target_device)
        if getattr(p, "_vllm_is_uva_offloaded", False):
            uva_offloaded_parameters.append(name)

```

```

try:
    yield module
finally:
    use_pin_memory = (
        is_pin_memory_available()
        and not envs.VLLM_WEIGHT_OFFLOADING_DISABLE_PIN_MEMORY
    )
    for name, p in module.named_parameters():
        if name in cpu_params:
            # 直接分配 pinned CPU 张量并拷贝，替代原来的
            # to("cpu") 非 pinned 拷贝；这样后续 H2D
            # 才能走非阻塞路径而不 stall。
            p.data = torch.empty_like(
                p.data, device="cpu", pin_memory=use_pin_memory
            ).copy_(p.data)

            # parameter is UVA offloaded, but was replaced with a new
            # device tensor: re-offload it to CPU using UVA, 同样用
            # 一次 pinned 拷贝完成，避免原先 to+cpu+pin 两次拷贝。
            if name in uva_offloaded_parameters and not getattr(
                p, "_vllm_is_uva_offloaded", False
            ):
                cpu_data = torch.empty_like(
                    p.data, device="cpu", pin_memory=use_pin_memory
                ).copy_(p.data)
                p.data = get_accelerator_view_from_cpu_tensor(cpu_data)
                p._vllm_is_uva_offloaded = True

```

vllm/model_executor/models/phi4mm_audio.py

Phi-4 音频 Conformer 编码器两处 streaming mask 的 H2D 改为 `async_tensor_h2d`（内部处理 pin + non_blocking），是音频编码路径上的直接收益点。

```

device = xs_pad.device
# 原实现: contiguous().to(device, non_blocking=True)
# 在源是普通 CPU 张量时并不会真正异步，会同步阻塞；
# async_tensor_h2d 统一处理“先 pin 再非阻塞拷贝”或
# 目标已在设备上时直接返回，避免握手阶段的 stall。
enc_streaming_mask = async_tensor_h2d(enc_streaming_mask.contiguous(), device)
xs_pad = xs_pad.to(device)

```

评论区精华

本 PR 来自 fork 分支，自动化 review 被禁用（claude[bot] 提示 maintainer 可手动触发 @claude review）；maintainer nooooop 直接 APPROVED，无阻塞意见。除 `/ci run` 触发命令（Buildkite #86072、#86083）外，没有实质性 review 讨论或未解决疑虑。唯一可见的设计权衡体现在 flashinfer.py 的代码注释中：移除常驻 pinned buffer 以规避 MRV2 下 step 间竞态，代价是每 step 多一次 CPU 侧 pin 快照拷贝。

- 暂无高价值评论线程

风险与影响

- 风险：核心热路径变更 (flashinfer.py) : `FlashInferMetadataBuilder` 是 V1 引擎 decode 每 step 必经路径。风险点：(a) 每 step 新增一次 CPU 侧 `pin_memory()` 快照拷贝 (`max_num_reqs + 1` 个 int32, 开销可忽略) ; (b) `PIN_MEMORY` 关闭时行为与 MRV2 现状等价, 不会更差; (c) cuda graph 模式下 `paged_kv_indptr.cpu` 被后续改写的问题由快照语义承接, 与旧双缓冲等价, 但缺少针对性回归测试。内存语义变化 (`model_loader/utils.py`) : CPU-resident 参数恢复为 pinned 内存后, 权重 offloading / sleep-wake 场景 host pinned 内存占用上升; `empty_like(p.data)` 假设 finally 中 `p.data` 仍是 GPU 张量, 依赖上下文顺序。多模型数值路径: Qwen2.5-VL、Qwen3-Omni、GLM-4.1V、Phi-4MM、ERNIE4.5V、Idefics2、Qwen3-ASR 的位置编码 / 掩码构造顺序被调整, 语义等价但无新增测试覆盖, 需依赖 CI 多模态正确性用例回归。
- 影响：用户侧: decode 阶段 GPU/CPU 重叠度提升, 降低因假异步 H2D 引起的 stall, 尤其影响 FlashInfer 后端与多模态 (视觉 / 音频) 输入处理路径。系统侧: V1 引擎 FlashInfer 元数据缓冲分配语义变化 (`paged_kv_indices` 常驻 GPU、移除双缓冲), 模型加载 / offloading 后 CPU 参数 pinned 化, pinned 内存占用小幅上升。团队侧: 与 #54292、#54293、#54295 构成同一“pin before non_blocking H2D”修复系列, 形成可复用的 `async_tensor_h2d` 模式。无 API/ 配置变更, 向后兼容。
- 风险标记: 核心路径变更, 缺少测试覆盖, pinned 内存占用上升, 跨模型多文件改动

关联脉络

- PR #53491 (前置 PR, 标题不在上下文中) : PR 正文明确说明本 PR 修复的阻塞实例由 #53491 暴露; 该 PR 为相关路径启用了异步 H2D 拷贝, 暴露了非 pinned 源导致的同步回退。
- PR #54292 [Perf] Pin CPU tensors before non_blocking H2D in three MM paths: 同一时期对 `mm_encoder_attention`、`kanana_v`、`kimi_k25_vit` 三处 MM 路径做相同的“H2D 前 pin”优化, 与本 PR 共用 `async_tensor_h2d` 模式。
- PR #54293 [Perf][KV Connector] Pin token_indices before non_blocking H2D in hf3fs helper: KV Connector 侧同主题修复: non_blocking H2D 前固定 host 张量, 属同一修复系列。
- PR #54295 [Perf][MLA Sparse] Pin req_id_per_token before non_blocking H2D on XPU and ROCm: XPU/ROCm 的 MLA Sparse 元数据路径同样在 H2D 前 pin, 与 flashinfer.py 的改动是同模式在其它注意力后端上的推广。